GUI Shaders Pack

by
Marcin Olszewski
(Frenzy Games)

July 3, 2021

1 About GUI Shaders Pack

GUI Shaders Pack contains advanced and multi-property shaders prepared for GUI. Most of shaders work with both Unity Text and Image components.

2 Where can be used

GUI Shaders Pack can be used with all latest Unity released versions. Most of shaders work on devices with shader model 3.0 (so with GLES 3+, DX 10+, OpenGL 4+), although there are some exceptions which work only on DX11+ (because they are using geometry and tesselation shader).

3 How to use it

To make shader (all are under path *FrenzyGames/GUI/**) work you need to attach it to material and this material attach to Image or Text. If material will be attached to text you should select *Is on Unity Text component* toggle. For example of shaders usage you can look at scene under path *Plugins/GUIShadersPack/Example/Scenes/**.

4 Shaders description

- **Dissolve** - shader for fading in/out GUI elements based on noise texture. Parameters:

  - *_MainTex* - source image in Image component
  - *_DissolveProgress* - progress of fade out
  - *_DissolveTex* - noise texture defining dissolve behaviour. UV is mapped in world space, so Tiling value should be rather small to see proper effect (usually values around 0.001-0.01 will be enough). Keep the same Tiling for multiple elements on scene so you will be able to make full GUI synchronised dissolve (of course if that effect is intended)
  - *Test dissolve tiling?* - select this to see how looks tiling of Dissolve Texture
  - *_BurnColor* - color of burn dissolve border
  - *_BurnRange* - width of burn border
  - *_BurnIntensity* - intensity of burn border (good for glow effect)
  - *Use BurTex gradient?* - select this to use gradient for burn instead of simple color
  - *_BurnTex* - gradient which defines burn color from begin to end of burn border. Check for example texture in *Plugins/GUIShadersPack/Example/Textures/burnGradient.* Make sure to set Wrap Mode in texture settings to Clamp.

- **Distortion** - shader for making *wavy* effect of GUI elements. Parameters:
  - *_MainTex* - source image in Image component
  - *_DistortionPower* - distortion power
  - *_DistortionTex* - bump map texture defining in which directions pixels will be pushed. UV is mapped in world space, so Tiling value should be rather small to see proper effect (usually values around 0.001-0.01 will be enough). Keep the same Tiling for multiple elements on scene so you will be able to make full GUI synchronised effect (of course if that effect is intended)
  - *_DistortionMoveDir* - x and y direction where Distortion Texture will be moved over time. Higher values mean faster movement
  - *Test distortion tiling?* - select this to see how looks tiling of Distortion Texture

- **GlowControl** - shader glow effect of GUI elements. To make it working fully you need to use bloom postprocess and have enabled HDR on camera. Parameters:

  - *_MainTex* - source image in Image component
  - *_GlowIntensity* - glow intensity (if you take value over 1.0, it will use HDR color values)
  - *Use selective glow?* - select this to enable selective glow. Glow is activated dependly on alpha value, if you set for example _AlphaTreshold = 0.7 than everything what have alpha over 0.7 will glow over 1.0 color value. Alpha value is scaled (and clamped to 0-1) by dividing by _AlphaTreshold, so values over exampled 0.7 will have alpha = 1.0
  - *_AlphaTreshold* - treshold over which alpha of texture is treated as glow factor

- **Hologram** - shader hologram effect. Parameters:
  - *_MainTex* - source image in Image component
  - *_GlowIntensity* - glow intensity (if you take value over 1.0, it will use HDR color values)
  - *_ColorMax* - color of visible lines (high pass)
  - *_ColorMin* - color of invisible lines (low pass)
  - *_LinesSetup* - x is density of lines and y is speed of move of lines

- **Materialization** - shader which generates small triangles (de)constructing GUI element. Requires DX11+(SM 5.0). Parameters:
  - *_MainTex* - source image in Image component
  - *_NoiseTex* - texture which defines moment and direction of dematerialization. R texture channel value works like in dissolve texture, so defines moment when part of image will be moved, GB are directions of part of element movement. Check out for *noise2 or noise3* textures from Example folder. UV is mapped in world space, so Tiling value should be rather small to see proper effect (usually values around 0.001-0.01 will be enough). Keep the same Tiling for multiple elements on scene so you will be able to make full GUI synchronised effect (of course if that effect is intended)
  - *Test noise tiling?* - select this to see how looks tiling of Noise Texture
  - *_Progress* - progress of dematerialization

  - *_FadeOutRange* - range of progress where dematerialization starts and ends. Lower value makes dematerialization, of single triangle, faster
  - *_MoveDistance* - distance which will be traveled by single triangle over _FadeOutRange
  - *_TriangeDensity* - density of tesselated triangles in element. Lower will make triangles more visible
  - *_BurnColor* - color of burn dissolve border
  - *_BurnIntensity* - intensity of burn border (good for glow effect)
  - *Use BurTex gradient?* - select this to use gradient for burn instead of simple color
  - *_BurnTex* - gradient which defines burn color from begin to end of burn border. Check for example texture in *Plugins/GUIShadersPack/Example/Textures/burnGradient.* Make sure to set Wrap Mode in texture settings to Clamp

- **Opaque** - shader which has disabled alpha blending and is set to Opaque mode. Great thing when you are going to optimize your GUI and gen rid of transparency blending. Parameters:
  - *_MainTex* - source image in Image component
  - *Use alpha clipping?* - makes clipping for alpha existing in texture. If you are going to optimize GUI for mobiles, try to avoid this, because even if this can be less expensive than alpha blending, still it can better to make multplie just opaque Images. Tip: for every change it's good to look at Unity Profiler results.
  - *_AlphaTreshold* - if alpha clipping is enabled, pixels are clipped under this value. Pixels over this value are fully visible

- **ColorCorrection** - shader for changing color, contrast, brightness, saturation and gamma on Images/Texts. Parameters:
  - *_MainTex* - source image in Image component
  - *Colorize?* - enables colorization. You can also just use Image color parameter instead of it.
  - *_Color* - multiplies image's color
  - *Use gammma correction?* - enables gamma correction
  - *_Gamma* - gammma value

  - *Use saturation correction?* - enables saturation correction
  - *_Saturation* - cotrast saturation
  - *Use brightness correction?* - enables brightness correction
  - *_Brightness* - brightness value
  - *Use contrast correction?* - enables contrast correction
  - *_Contrast* - contrast value

- **Blured** - shader for bluring image by Gaussian Blur (cross step, so sampling goes with vector (1,1) and (-1,-1). Parameters:
  - *_MainTex* - source image in Image component
  - *Colorize?* - enables colorization. You can also just use Image color parameter instead of it.
  - *_Color* - multiplies image's color
  - *_BlurRadius* - radius of blur in texture pixels count
  - *_BlurIntensity* - how much blur will be used (value under 1 makes image more like basic texture)
  - *Blur samples* - count of Gaussian Blur steps, more is also more texture sampling (so more expensive for GPU)

- **Bluring** - shader for translucent bluring GUI under image by Gaussian Blur (cross step, so sampling goes with vector (1,1) and (-1,-1). It uses GrabPass so it may be too expensive for low-end devices although it's still faster than full Gaussian Blur (so vertical + horizontall instead of just cross step). Parameters:
  - *_MainTex* - source image in Image component, multiplies color under image. Alpha of image says where bluring will be visible (you can alos manipulate with Image color).
  - *_BlurRadius* - radius of blur in screen pixels count
  - *_Translucency* - how much blur will be used
  - *Blur samples* - count of Gaussian blur steps, more is also more texture sampling (so more expensive for GPU)

- **Tween** - shader for move, scale and rotate operations on Image. Great thing for animations and keeping good their good performance. As in other shaders, if you use the same material for multiple GUI elements, you can animate multiple elements by just changing this single material. Parameters:

- *_MainTex* - source image in Image component.
- *Colorize?* - enables colorization. You can also just use Image color parameter instead of it.
- *_Color* - multiplies image's color
- *_OperationsPivot* - pivot point which will have impact on scale and rotate operations. If you have selected Image with fold-outed material you should see _OperationsPivot rect to be able to easily select where it is.

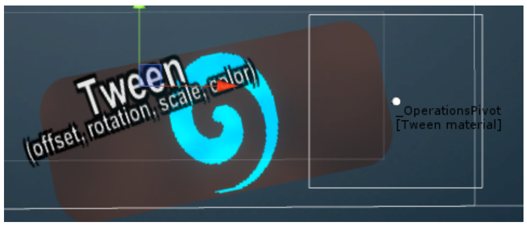

Figure 1: Pivot point visible on scene view

- *Move?* - enables _Offset use
- *_Offset* - final offset
- *Rotate?* - enables rotation
- *_Rotation* - final rotation around _OperationsPivot
- *Scale?* - enables scale
- *_Scale* - final scale from _OperationsPivot
- *Use _Progress to move from zero point to final?* - enables progressive changes to set Tween values (offset, rotation and scale). Great thing if you want to animate Image/Text to/from expected position from/to basic one
- *_Progress* - progression from basic Image/Text values to final offset/rotation/scale

• **Parallax** - scripted shader for parallax effect which can make that GUI will look like has some depth and UI elements' inside can rotate with cursor or for example rotation of phone. About idea of this effect you can read here `http://graphics.cs.brown.edu/games/SteepParallax/index.html`. Requires GUIParallaxShaderSupport script. Parameters:

  - *SCRIPT:Update in edit mode* - values will update also in edit mode when object is selected.
  - *SCRIPT:Will object change during runtime* - select if image will change its position or size.
  - *SCRIPT:Look dir control* - type of what will control the parallax mapping look direction (can be manual/custom script or controlled by camera/mouse). *Local* types use local position of object to calculate look dir, *global's* use the center of screen.
  - *SCRIPT:Look dir factor* - multiplier for final look direction. Useful if you want to intense one of axis or invert it.
  - *SCRIPT:Look dir* - look direction passed to parallax mapping shader.
  - *SCRIPT:Observer* - some of look control types require camera to know from what values should be calculated.
  - *_HeightMap* - height map of the image, used to calculate final offset of pixels.
  - *_HeightScale* - says how big offset will be made by parallax taken from height map.
  - *_ParallaxSamples* - count of steps on height map texture used to make proper main texture pixels offset. Note that more looks better but also are more expensive (usually values like 5-10 are enough).
  - *Clamp patallax offset* - cuts offset of parallax when it goes out of texture tiling. Note that if texture is in atlas than it will not lock going out of basic bounds.

- **Scripted shaders** - shaders which work simillary to above shaders but are under path *FrenzyGames/GUI/Scripted/** and require to have *GUIShaderSupport*(or another mentioned in script properties) script attached into Image/Text game object. Please note that if you want to see proper effect in Edit mode you should select in script *UpdateInEditMode* field (this will also force script to update attached to Image material in edit mode, so unselect it when you stop changing settings for effect).

5 Contact

If you need any support or you would like to share with us what do you think about this package, feel free to write on support@modev.com.pl . Also if you need some additional parameters which can be used write to me too - maybe I'll make this and put it into next update.

It would be great if you could rate this asset on Asset Store. This will be good sign for me that package is needed and should be still expanded.